US-PAT-NO: 5410704

DOCUMENT-IDENTIFIER: US 5410704 A

TITLE: Table modifiable edit functions with order-effective edit rules

---------- KWIC ---------

Application Filing Date - AD (1):
19930301

Drawing Description Text - DRTX (6):
FIG. 4 Shows a block diagram illustrating the relationship of the **forms manager** to the application program and database.

Detailed Description Text - DETX (24):
Forms Manager

Detailed Description Text - DETX (25):
FIG. 4 shows a block diagram illustrating the relationship of the **forms manager** to the application program and database. A **forms manager** 111 is used as an interface, or buffer, among and between the input, output, and system applications and databases.

Detailed Description Text - DETX (26):
An input device 112 is coupled to **forms manager** 111. Input device 112 is illustrated as a keyboard with a joy stick but may be any type of input device. In general, the communication is from input device 112 to **forms manager** 111. However, there may be a need for **forms manager** 111 to operate visual or acoustic devices on input device 112. For this reason, there may be some communication from **forms manager** 111 to input device 112.

Detailed Description Text - DETX (27):
Also coupled to **forms manager** 111 is an operator display module 113. Display 113 is typically a cathode ray tube, but may be any of numerous types of displays such as the various types of flat panel displays. Communication is generally from **forms manager** 111 to display 113. However, a touch sensitive screen or other type of input device may be incorporated in the system requiring communication from display 113 to **forms manager** 111.

Detailed Description Text - DETX (28):
Forms manager 111 also interacts with the application programs 114 and the

databases. The databases, for this particular example, are broken into a general database 115 and a specific database 116. It should be understood that while the database may be functionally separated, both may be contained in the same physical location.

Detailed Description Text - DETX (31):
In operation, objects are entered through input device 112 to the **forms manager**. This input is checked and stored in database 116 for the particular patient. When a form is to be displayed, **forms manager** 111 obtains a list of objects to be displayed. One source of objects is application program 114. The **forms manager** then retrieves the display tiles associated with the objects from database 115. The data to be placed in the tiles is then obtained from database 116. Using these inputs, **forms manager** 111 develops a form which is displayed on display 113.

Detailed Description Text - DETX (34):
The application software architecture is a way to specify action-oriented applications. The **forms manager** 100 translates user actions into "events" and sends the events to the graph manager 101.

Detailed Description Text - DETX (36):
The user performs actions. The **forms manager** 100 generates events based on the user's actions. The graph manager 101 catches the events generated by the **forms manager** 101.

Detailed Description Text - DETX (98):
A list procedure is used to implement a list whose list items vary during run-time. The list procedure selects items from a master list based on criteria specified in the body of the list procedure and passes those items to the **forms manager** for display.

Detailed Description Text - DETX (111):
There are two controlling processes, the **forms manager** and the graph manager. The **forms manager** handles all user I/O, as well as the details of exactly what appears on the screen, including such things as pop-up windows with lists of choices for the user. The **forms manager** also translates user actions into events and sends those events to the graph manager.

Detailed Description Text - DETX (112):
The graph manager receives from the **forms manager** events and which attributes those events are associated with (one attribute per event). The graph manager screens incoming events, passing some to the applications and acting on others. Based on events received, the graph manager calls appropriate procedures to handle default values, validation of new values (edits), and calculation of values.

Detailed Description Text - DETX (113):
Both the **forms manager** and the graph manager determine which DECL's procedures to invoke by using a look-up table in a database.

Detailed Description Text - DETX (117):
The default procedure determines the default value for the attribute (possibly based on database attribute values, other screen values, etc), gives it to the **forms manager** to display on the screen, and returns.

Detailed Description Text - DETX (120):
If any one edit procedure fails, the graph manager empties the event queue (where all incoming events are kept until the graph manager can process them). The edit procedure that failed notified the **forms manager** of the error and supplied the **forms manager** with an error message to display.

Detailed Description Text - DETX (124):
Lists and the **Forms Manager**

Detailed Description Text - DETX (125):
When the user selects a field on the screen that is enterable (i.e. the user is allowed to enter data), the graph manager looks up the field's associated attribute in an attribute table to determine if there is a list procedure associated with the attribute. If there is, the **forms manager** calls the list procedure, which returns a list of objects. The **forms manager** then displays the list in a pop-up window so that the user can select an item from it for the selected field.

Detailed Description Text - DETX (126):
The **forms manager** generates other special events, such as cross-edit events for cross-field edit procedures (which appear in the attribute table in the database under a column for cross-edit procedures).

Detailed Description Text - DETX (138):
FIG. 14 shows a "Dialog Node" 241. Dialog nodes interact with the **Form Manager** to control data display and entry. When dialog node 241 is invoked (via SC-node-invoke), it retrieves the definition of its processing (via QR-node-d), and also receives a set of import attributes and the name of the invoking arc (via SC-node-invoke). It may use a variety of tables to specify options in how it controls the dialog (via QR-app-config-d).

Detailed Description Text - DETX (139):
Dialog nodes make requests to the **Form Manager** to display data on various forms (via CF-interaction-control and interaction-results) and receive notification of various events (e.g. field selected) via CF-filtered-input-events. The dialog nodes specify what actions are to be carried out for each of these events (e.g. change menu definitions, proceed to

a particular field), again via CF-interaction-control.

Detailed Description Text - DETX (161):
For example, the division of functionality between the graph manager and the **forms manager** may be altered in different ways, so that one may perform functions described herein as performed by the other, and vice versa.

Detailed Description Text - DETX (178):
Dialog node=interacts with the **forms manager** to control data display and entry. Makes requests to the **forms manager** to display data on various forms and receive notification of various events.

Detailed Description Text - DETX (181):
Forms manager=translates user actions into events and sends those events to the graph manager.

Detailed Description Text - DETX (227):
If any of the previous edits failed (this takes care of case where **forms manager** sends a slew of cross.sub.-- edit events followed by an ok, sign, or more event)